

Implementing Distance Vector Routing Protocol

Distance Vector Routing Protocol:

In this programming assignment, we have implemented Distance vector routing protocol where each node exchanges the routing information with directly connected neighbors. Routers select the route with the lowest cost to each possible destination and add this to their own routing tables. These neighbors propagate the information to their neighbors hop by hop until information from all routers has spread throughout the entire internetwork.

Distance vector routing protocol is based on Bellman-Ford algorithm in which every router maintains a database with one entry for each possible destination on the network.

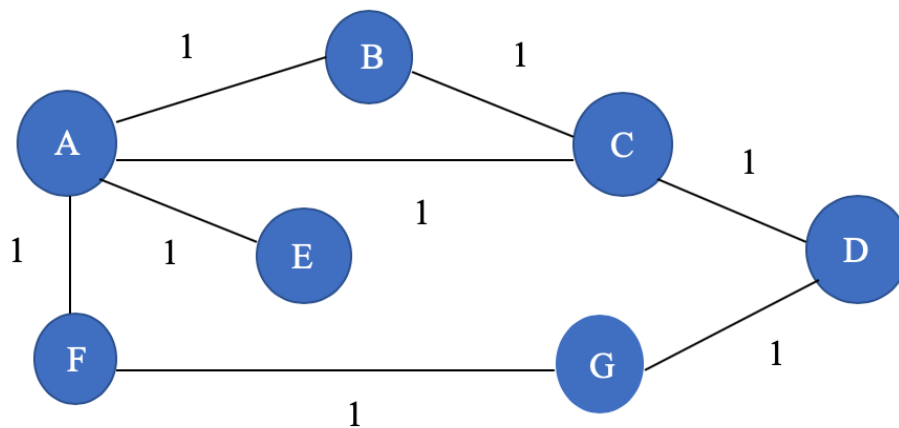


Fig. 1

Following steps are followed in Distance Vector Routing Protocol:

- Every node sends a message to its directly connected neighbors containing its personal list of distance. (for example, A sends its information to its neighbors B, C, E, and F)
- If any of the recipients of the information from A find that A is advertising a path shorter than the one they currently know about, they update their list to give the new path length and note that they should send packets for that destination through A. (node B learns from A that node E can be reached at a cost of 1; B also knows it can reach A at a cost of 1, so it adds these to get the cost of reaching E by means of A. B records that it can reach E at a cost of 2 by going through A.)
- After every node has exchanged a few updates with its directly connected neighbors, all nodes will know the least-cost path to all the other nodes.

- In addition to updating their list of distances when they receive updates, the nodes need to keep track of which node told them about the path that they used to calculate the cost, so that they can create their forwarding table. (for example, B knows that it was A who said " I can reach E in one hop" and so B puts an entry in its table that says " To reach E, use the link to A.)

The DV calculation is based on minimizing the cost to each destination

$D_x(y)$ = Estimate of least cost from x to y

$C(x,v)$ = Node x knows cost to each neighbor v

$D_x = [D_x(y): y \in N]$ = Node x maintains distance vector

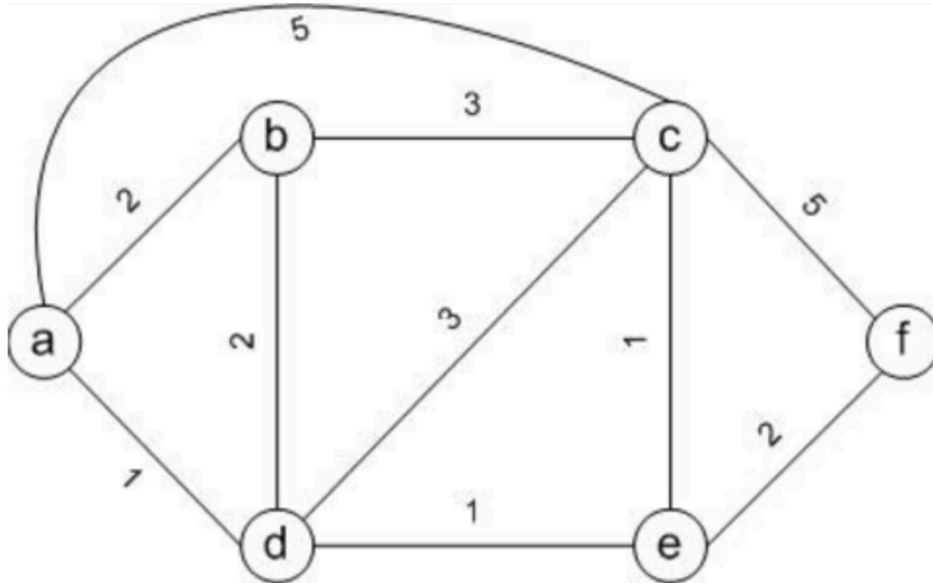
Node x also maintains its neighbors' distance vectors

– For each neighbor v, x maintains $D_v = [D_v(y): y \in N]$

Implementation of Distance Vector Routing Protocol:

- In this implementation, the input to the program is the set of directly attached links and their costs. The program at each host doesn't know the complete network topology
- The program takes a text file as the input which describes the set of directly attached links along with the costs associated with them. The first line of the text input file is a single number which denotes the number of directly attached links while the rest of the lines gives the cost between two different nodes. The input file is in the .dat format. The output of this project implementation is the cost as well as the next hop of the shortest path between the various network nodes
- As per the requirement, this project is a variation to the original Distance Vector Routing Protocol where each host sends out the routing information to its neighbors at a certain frequency (once every 15 seconds) irrespective of whether the information has changed since the last announcement.
- The host recomputes its distance vector as well as the routing table before passing the information to its neighbors.
- This project implementation is also able to handle link cost changes along with the recursive update problem successfully.

Below example is being provided in the implementation but it can be updated and will work for any given value. Every node information is provided in individual .dat file.



Terminal output for each router:

Initial cost information of connected neighbors

Node A:

```
a node x
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
output number 1
shortest path a-b: the next hop is b and the cost is 2.0
shortest path a-c: the next hop is c and the cost is 5.0
shortest path a-d: the next hop is d and the cost is 1.0
```

Node B:

```
b node x
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
output number 1
shortest path b-a: the next hop is a and the cost is 2.0
shortest path b-c: the next hop is c and the cost is 3.0
shortest path b-d: the next hop is d and the cost is 2.0
```

Node C:

```
c node x
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
output number 1
shortest path c-a: the next hop is a and the cost is 5.0
shortest path c-b: the next hop is b and the cost is 3.0
shortest path c-d: the next hop is d and the cost is 3.0
shortest path c-e: the next hop is e and the cost is 1.0
shortest path c-f: the next hop is f and the cost is 5.0
```

Node D:

```
d node x
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
output number 1
shortest path d-a: the next hop is a and the cost is 1.0
shortest path d-b: the next hop is b and the cost is 2.0
shortest path d-c: the next hop is c and the cost is 3.0
shortest path d-e: the next hop is e and the cost is 1.0
```

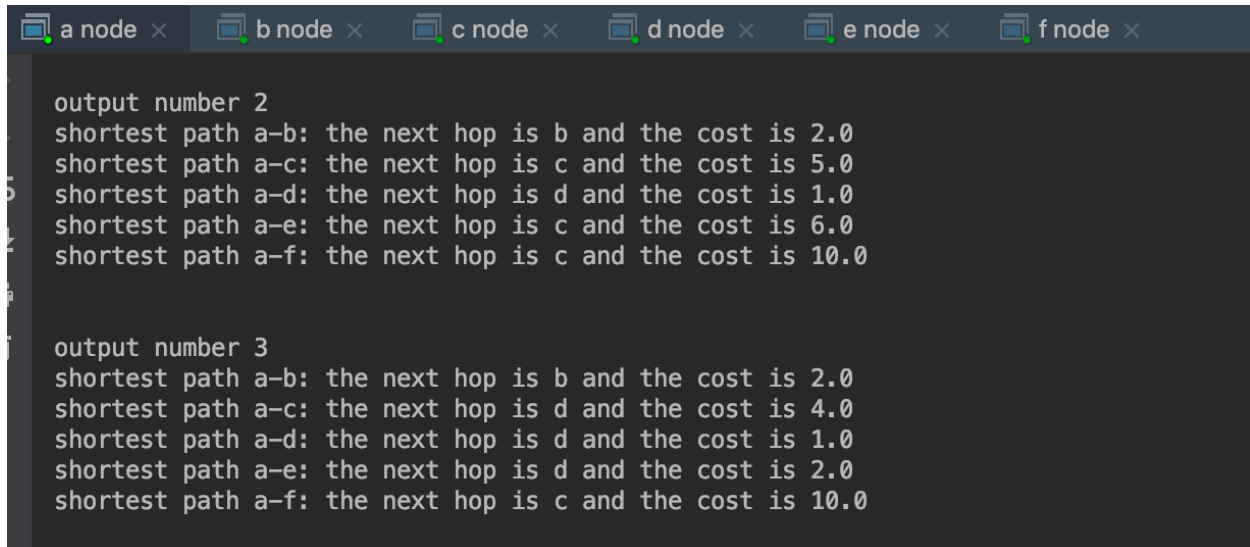
Node E:

```
e node x
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
output number 1
shortest path e-c: the next hop is c and the cost is 1.0
shortest path e-d: the next hop is d and the cost is 1.0
shortest path e-f: the next hop is f and the cost is 2.0
```

Node F:

```
f node x
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
output number 1
shortest path f-c: the next hop is c and the cost is 5.0
shortest path f-e: the next hop is e and the cost is 2.0
```

As each host propagates the cost information to their neighbors hop by hop, we can see that each node has routing information of every other node present in the network

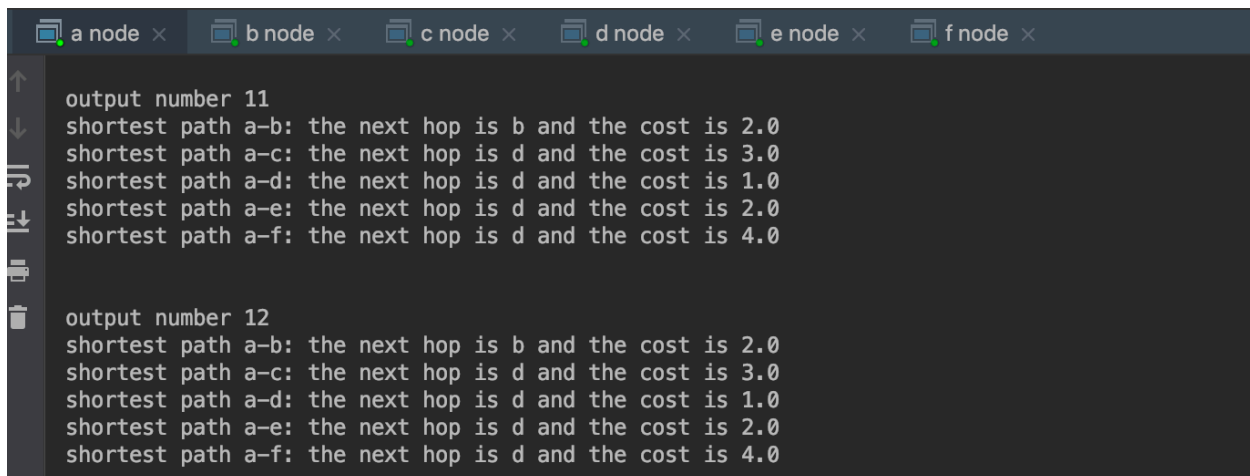


```
a node x b node x c node x d node x e node x f node x

output number 2
shortest path a-b: the next hop is b and the cost is 2.0
shortest path a-c: the next hop is c and the cost is 5.0
shortest path a-d: the next hop is d and the cost is 1.0
shortest path a-e: the next hop is c and the cost is 6.0
shortest path a-f: the next hop is c and the cost is 10.0

output number 3
shortest path a-b: the next hop is b and the cost is 2.0
shortest path a-c: the next hop is d and the cost is 4.0
shortest path a-d: the next hop is d and the cost is 1.0
shortest path a-e: the next hop is d and the cost is 2.0
shortest path a-f: the next hop is c and the cost is 10.0
```

As observed in the above screenshot, initially each node has information of only it's connected neighbor but as each node begins to advertise and propagate information to their neighbors, every node has routing information of every connected node. With addition of more nodes and gathering of routing information, minimizing the cost to each destination is achieved.



```
a node x b node x c node x d node x e node x f node x

output number 11
shortest path a-b: the next hop is b and the cost is 2.0
shortest path a-c: the next hop is d and the cost is 3.0
shortest path a-d: the next hop is d and the cost is 1.0
shortest path a-e: the next hop is d and the cost is 2.0
shortest path a-f: the next hop is d and the cost is 4.0

output number 12
shortest path a-b: the next hop is b and the cost is 2.0
shortest path a-c: the next hop is d and the cost is 3.0
shortest path a-d: the next hop is d and the cost is 1.0
shortest path a-e: the next hop is d and the cost is 2.0
shortest path a-f: the next hop is d and the cost is 4.0
```

```
a node x b node x c node x d node x e node x f node x

output number 3
shortest path b-a: the next hop is a and the cost is 2.0
shortest path b-c: the next hop is c and the cost is 3.0
shortest path b-d: the next hop is d and the cost is 2.0
shortest path b-e: the next hop is d and the cost is 3.0
shortest path b-f: the next hop is c and the cost is 8.0

output number 4
shortest path b-a: the next hop is a and the cost is 2.0
shortest path b-c: the next hop is c and the cost is 3.0
shortest path b-d: the next hop is d and the cost is 2.0
shortest path b-e: the next hop is d and the cost is 3.0
shortest path b-f: the next hop is c and the cost is 6.0

output number 5
shortest path b-a: the next hop is a and the cost is 2.0
shortest path b-c: the next hop is c and the cost is 3.0
shortest path b-d: the next hop is d and the cost is 2.0
shortest path b-e: the next hop is d and the cost is 3.0
shortest path b-f: the next hop is d and the cost is 5.0
```

References:

- <https://www.geeksforgeeks.org/computer-network-routing-protocols-set-1-distance-vector-routing/>
- https://en.wikipedia.org/wiki/Distance-vector_routing_protocol#Count-to-infinity_problem
- https://www.cs.bu.edu/fac/byers/courses/791/F99/scribe_notes/cs791-notes-990923.html